

Application for United States Letters Patent

for

**Adaptive Equalization using a Conditional Update Sign-Sign Least Mean Square
Algorithm**

Ganesh Balamurugan

Bryan K. Casper

James E. Jaussi

Stephen R. Mooney

Prepared by:

Seth Z. Kalson
Intel Corporation
2200 Mission College Blvd.
Santa Clara, CA 95052-8119

"Express Mail" mailing label number: EV339923211US

Date of Deposit: 9/10/2003

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Commissioner for Patents, Washington, D. C. 20231

Anne Collette

(Typed or printed name of person mailing paper or fee)

Anne Collette

(Signature of person mailing paper or fee)

9/10/2003

(Date signed)

Field

[0001] The present invention relates to communication circuits, and more particularly, a circuit for adaptive equalization of a communication channel.

Background

[0002] Fig. 1 provides a high-level abstraction of a portion of a computer server or system, where microprocessor 102 resides on board 104 and communicates with memory 106 on board 108. The communication is by way of striplines on backplane 110. Backplane 110 is connected to boards 104 and 108 by connectors 112. Not shown in Fig. 1 are other memory units and microprocessors, where the various microprocessors and memory units may communicate to one another so as to access or write data and instructions.

[0003] Communication of signals over backplane 110 may be modeled by transmission line theory. Often, the signaling is based upon differential signaling, whereby a single bit of information is represented by a differential voltage. For example, Fig. 2a shows drivers 202 and 204 driving transmission lines 206 and 208, respectively. For differential signaling, drivers 202 and 204 drive their respective transmission lines to complementary voltages. Typical curves for the node voltages at nodes n1 and n2 for a bit transition are provided in Fig. 2b, where the bit transition is indicated by a dashed vertical line crossing the time axis. The information content is provided by the difference in the two node voltages.

[0004] For short-haul communication, such as for the computer server in Fig. 1, the signal-to-noise ratio is relatively large. If the transmission lines are linear, time-invariant systems having a bandwidth significantly greater than that of the transmitted signal, then a relatively simple receiver architecture may be employed to recover the transmitted data. Such a receiver is abstracted by comparator 210, which provides a logic signal in response to the difference in the two received voltages at ports 212 and 214.

[0005] However, every transmission line has a finite bandwidth, and for signal bandwidths that are comparable to or exceed the transmission line (channel) bandwidth, intersymbol interference may present a problem. Furthermore, actual transmission lines may have dispersion, whereby different spectral portions of a signal travel at different speeds. This may result in pulse spreading, again leading to intersymbol interference. As

a practical example, for high data rates such as 10Gbps (Giga bits per second), the transmission lines used with backplanes or motherboards are such that intersymbol interference is present.

[0006] Channel equalization is a method in which one or more filters are employed to equalize the channel to help mitigate intersymbol interference. These filters may be sampled-data (discrete-time) filters, where a time index t is a discrete variable, or they may be continuous-time filters, where the time index is a continuous variable. Many channel equalizers are realized by a Finite Impulse Response (FIR) filter employed at the receiver. A FIR is a sampled-data filter. Fig. 3 is an abstraction of a FIR filter structure, where $x(t)$ is the input (received) signal to the FIR filter and $z(t)$ is the filtered output. In the case of differential signaling, $x(t)$ and $z(t)$ may be viewed as representing differential signals. The filter impulse response is represented by a n dimensional vector denoted as \bar{h} , where the filter weights are the components of this vector and are denoted by $[\bar{h}]_i$, $i = 0, 1, \dots, n - 1$. Multipliers 302 and summer 304 may be realized in the analog or digital domain. For the embodiments described in these letters patent, multipliers 302 and summer 304 are realized in the analog domain.

[0007] In many adaptive equalization schemes, the filter vector is updated during a training time interval, and then remains fixed for some period of time. During training, a known sequence is transmitted over a communication channel to the receiver, and the filter vector is synthesized during the training time interval. Many algorithms have been developed to synthesize the filter vector. The well-known LMS (Least Mean Square) algorithm is an iterative technique based upon the method of steepest descent (gradient) to minimize a squared error.

[0008] The LMS algorithm may be written as the following iterative procedure performed during the training time interval:

$$\bar{h}(t + 1) = \bar{h}(t) + \mu [Kd(t) - z(t)]\bar{x}(t),$$

where $\bar{x}(t)$ is a n dimensional received data vector with components given by

$[\bar{x}(t)]_i = x(t - i)$ for $i = 0, 1, \dots, n - 1$, μ is a positive weight determining the filter “memory” or “window size” and may be viewed as the step-size in the steepest descent method, $d(t)$ represents the known transmitted data during the training time interval (the

training sequence), and K is a positive scale factor. The above iteration is performed during a training time interval $t = 1, \dots, T$, where an initial $\bar{h}(0)$ is chosen and at the end of the training time interval, the filter vector \bar{h} is set equal to $\bar{h}(T)$, i.e., $\bar{h} = \bar{h}(T)$.

[0009] In many analog implementations the filter weights assume discrete values limited to some fixed range, and the scale factor K takes into account this finite range of the filter weights as well as practical implementations of the filtering. For example, for bipolar differential signaling the known training sequence $d(t)$ may take on either of the values V_{CC} or $-V_{CC}$, where V_{CC} is a supply voltage, but in practice the filtered output $z(t)$ is always in magnitude less than the supply voltage. In this case, $K < 1$.

[0010] To simplify the computations needed to synthesize the filter vector, the so-called sign-sign-LMS algorithm has been used:

$$\bar{h}(t+1) = \bar{h}(t) + \mu \operatorname{sgn}\{[Kd(t) - z(t)]\} \operatorname{sgn}\{\bar{x}(t)\},$$

where $\operatorname{sgn}\{ \cdot \}$ is the sign function. The sign-sign-LMS algorithm, although widely used due to its simplicity of implementation, has been found to have some undesirable properties when used in adaptive high-speed equalizers with relatively low to moderate length words (e.g., four to six bits) for the filter weights. Because the probability of filter weight update during adaptation is high, there is a significant amount of residual noise in the filter weights, even after convergence of the algorithm. This residual noise may be reduced by choosing a longer window (smaller μ), but this increases the convergence time, or in other words, the training time interval. Another disadvantage found in many instances is that the converged filter weights are relatively sensitive to the scale factor K , whose optimum value has been found to be difficult to determine.

Brief Description of the Drawings

[0011] Fig. 1 is a high-level abstraction of a server system, illustrating signaling between boards via a backplane.

[0012] Fig. 2a illustrates differential signaling on two transmission lines.

[0013] Fig. 2b shows typical voltage curves representing differential signaling on the transmission lines of Fig. 2a.

[0014] Fig. 3 illustrates a finite impulse response (FIR) filter, or tap delay-line.

- [0015] Fig. 4 illustrates a circuit at a high-level functional level for implementing adaptive equalization as taught in these letters patent.
- [0016] Fig. 5 illustrates a circuit at a high-level functional level for implementing multiplication in the analog domain for the circuit of Fig. 4.
- [0017] Fig. 6 is a more detailed illustration of the circuit of Fig. 5 for the special case of 2-bit multiplier weights.
- [0018] Fig. 7 is a circuit for a differential latch that may be used in the circuit of Fig. 4.
- [0019] Fig. 8 is a flow diagram illustrating a method for calibrating the scale factor used in the circuit of Fig. 4.
- [0020] Fig. 9 illustrates a circuit at a high-level functional level for implementing the calibration method of Fig. 8 for the adaptive equalizer of Fig. 4.
- [0021] Fig. 10 illustrates a circuit at a high-level functional level for implementing the calibration method of Fig. 8 for a wide class of adaptive equalizers.

Description of Embodiments

- [0022] Fig. 4 provides an abstraction of an embodiment of the present invention. The embodiment of Fig. 4 performs the following adaptive update:

$$\bar{h}(t+1) = \bar{h}(t) + \mu[\operatorname{sgn}\{d(t)\} - \operatorname{sgn}\{z(t) - Kd(t)\}]\operatorname{sgn}\{\bar{x}(t)\}, \quad (1)$$

which may be termed the conditional update sign-sign LMS algorithm. Equalizer 402 is a FIR filter with filter weights $[\bar{h}(t)]_i$, $i = 0, 1, \dots, n-1$, and provides the filtered output

$$z(t) = \sum_{i=0}^{n-1} [\bar{h}(t)]_i x(t-i). \text{ Each filter weight is represented by a set of discrete voltages}$$

taking on either 0 or V_{cc} (LOW or HIGH), so that each filter weight may be viewed as a discrete variable in the digital domain represented by a finite number of bits. However, the filtering is performed in the analog domain so that the filtered output $z(t)$ is an analog voltage signal. If differential signaling is employed, then both analog voltages $x(t)$ and $z(t)$ are differential signals. An example of FIR 402 at the circuit level will be described later. Not shown (for simplicity) in Fig. 4 are clock signals and latches to perform the various multiplications, summations, comparisons, and symbol generation at discrete times $t = 0, 1, \dots, T$ during the training sequence.

[0023] Data symbol generator 404 provides a sequence of voltages in the digital domain representing $\text{sgn}\{d(t)\}$. Because $\text{sgn}\{\}$ obviously assumes only one of two values, the sequence of voltages representing $\text{sgn}\{d(t)\}$ may be viewed as a sequence of binary values 0 or 1, which may be stored in a memory unit or generated by a finite state machine. Data symbol generator 404 also provides a sequence of voltages in the digital domain representing $d(t)$. In digital communications, the transmitted symbols range over a finite set, so that (remembering that $d(t)$ is the transmitted training sequence) each $d(t)$ may be represented by a finite number of bits. These bits may be stored in memory, or generated by a finite machine. In the case of bipolar signaling, only one bit of information is needed to represent $d(t)$, so the same sequence used to represent $\text{sgn}\{d(t)\}$ may also represent $d(t)$. Circuits for data symbol generator 404 are relatively straight forward to synthesize. For example, in the bipolar signaling case, it is well known that the use of feedback shift registers may be used to generate the binary bits representing the data $d(t)$.

[0024] Symbol-to-voltage converter 406 converts the digital domain voltages representing $d(t)$ into analog voltages representing $d(t)$. For example, consider the case in which differential bipolar signaling is employed. In this case, only one bit of information, for example a voltage 0 or V_{CC} , is needed to represent $d(t)$ and is provided to the input of symbol-to-voltage converter 406, and two voltages are used to represent the differential output voltage of symbol-to-voltage converter 406. For example, in this case, symbol-to-voltage converter 406 may provide the differential voltage $[0, V_{CC}]$ in response to one of its input voltages, say 0, and the differential voltage $[V_{CC}, 0]$ in response to the input voltage V_{CC} . Such a circuit is of course obvious, and for the more general case, circuits for symbol-to-voltage converter 406 are straightforward to implement.

[0025] Multiplier 408 multiplies the analog voltage $d(t)$ by the negative of the scale factor, $-K$. This multiplication is performed in the analog domain, but just as for the filter $\bar{h}(t)$, the scale factor is represented by a set of discrete voltages taking on the values 0 or V_{CC} (LOW or HIGH), so that K is represented by a finite number of bits.

Summer 410 adds the analog voltage $-Kd(t)$ to the filtered output $z(t)$, which is provided to the input of comparator 412. The multiplier 408 and summer 410 have the same structure as a filter tap, so that an example circuit for multiplier 408 and summer 410 is described later in connection with FIR 402.

[0026] Comparator 412 provides a logic output signal indicative of the difference of its two input signals (each input signal is part of a differential voltage in the differential signaling case), so that it outputs $\text{sgn}\{z(t) - Kd(t)\}$ as a discrete voltage 0 or V_{cc} (LOW or HIGH) in the digital domain. (It is immaterial whether comparator 412 evaluates $\text{sgn}\{0\}$ as a LOW or HIGH voltage.) Comparator 412 also includes a latch for latching its output voltage. (For simplicity, clock signals are not shown in Fig. 4.) Multiplier 414 and summer 416 effectively subtract $\text{sgn}\{z(t) - Kd(t)\}$ from $\text{sgn}\{d(t)\}$, and multiplier 418 multiplies this difference by the filter memory parameter μ , which is then provided as an operand to multiplier 424. The data vector $\bar{x}(t)$ is provided to comparator 426, which outputs the vector $\text{sgn}\{\bar{x}(t)\}$ as an operand to multiplier 424. (Comparator 426 also includes a latch.) The data vector $\bar{x}(t)$ may be provided by FIR 402 because this vector is also used by FIR 402 to perform filtering. The values of $\bar{x}(t)$ may be obtained by samplers, not shown for simplicity. Multiplier 424 multiplies its two operands together, so that each component of $\text{sgn}\{\bar{x}(t)\}$ is multiplied by the scalar $\mu[\text{sgn}\{d(t)\} - \text{sgn}\{z(t) - Kd(t)\}]$, and this result is provided to summer 420. Summer 420, delay element 422, and the feedback from delay element 422 to summer 420 perform the desired iteration on the filter weights as expressed by equation (1).

[0027] At the end of the training sequence, the filter weights for equalization are given by $\bar{h}(T)$, in which case FIR 402 may be disconnected from the circuit so as to provide the equalized output $z(t)$ for $t > T$.

[0028] Multipliers 414 and 418, summers 416 and 420, and unit delay shift 422 operate in the digital domain, whereas the multiplications and summation performed within FIR 402, as well as summer 410 and multiplier 408, are performed in the analog domain. It should be remembered that the multiplication weights $\bar{h}(t)$ and K in Fig. 4 are discrete-valued signals (i.e., digital signals) represented by a finite number of bits

(HIGH and LOW voltages). Although $d(t)$ is a discrete-valued signal due because of the nature of digital communication, it is nevertheless viewed as an analog signal because it is a differential voltage input to a voltage-to-current converter when performing the multiplication indicated by multiplier 408, as described later. Circuits for performing operations in the digital domain (i.e., logic circuits) are well known, and need not be described here. A circuit implementation for performing the analog filtering (multiplication and summation) in Fig. 4 is now described below.

[0029] A circuit to implement multiplication in the analog domain utilizing differential signals is shown in Fig. 5, where the differential signal voltage $x(t)$, represented by the two voltages $\{x_+(t), x_-(t)\}$, is effectively multiplied by $[\bar{h}]_i$, the i^{th} component of the filter vector \bar{h} . Multiplication by $[\bar{h}]_i$ also involves its bit-wise logical complement, denoted as $[\bar{h}]_i^c$, which is easily generated. The multiplier indicated by Fig. 5 comprises voltage-to-current converter 502 and current steering DAC (digital-to-analog converter) 504. Voltage-to-current converter 502 provides differential current $\{I_+, I_-\}$ such that the difference $I_+ - I_-$ is proportional to $V_+ - V_-$, that is, $I_+ - I_- = \alpha(V_+ - V_-)$, where α is a transconductance scalar. Current steering DAC 504 steers currents $\gamma[\bar{h}]_i I_-$ and $\gamma[\bar{h}]_i I_+$, respectively, to loads 506 and 508, and shunts the rest of the current (if any) to ground. The scalar γ is given by $\gamma = 1 / \max\{\text{abs}([\bar{h}]_i)\}$. This scalar is introduced because current steering DAC 504 can at most steer all of currents I_+ and I_- to, respectively, loads 508 and 506. With this scalar, the absolute value of $\gamma[\bar{h}]_i$ will always be bounded by one. In practice, loads 506 and 508 may be realized by active devices, but for simplicity they are represented as resistive loads in Fig. 5.

[0030] The voltage developed at nodes 510 and 512 are, respectively, $Z_L \gamma[\bar{h}]_i I_-$ and $Z_L \gamma[\bar{h}]_i I_+$, where Z_L is the impedance of loads 506 and 508. The difference in voltages developed at nodes 512 and 510 is given by $\alpha Z_L \gamma[\bar{h}]_i (V_+ - V_-)$. The difference in voltages developed at node 512 and 510 is seen to be proportional to the desired multiplication $[\bar{h}]_i (V_+ - V_-)$ where the proportionality is the dimensionless scalar $\alpha Z_L \gamma$.

This dimensionless scalar is not of theoretical concern because it is taken into account by the scale factor K when performing the filter weight update.

[0031] An example of voltage-to-current converter **502** and current steering DAC **504** at the circuit level is shown in Fig. 6. For simplicity, only a 2-bit current steering DAC is shown, so that the filter weights are assumed to be represented by two-bit words each. It should be straightforward how any number of bits may be implemented. In Fig. 6, b_1 and b_0 are voltages representing binary variables, where a HIGH voltage represents the value 1 and a LOW voltage represents the value 0. Incurring the typical, but slight abuse of notation, we let b_1^c and b_0^c also represent binary variables, where b_1^c and b_0^c are, respectively, the logical complements of b_1 and b_0 . It will be clear from context whether these variables represent voltages or binary values, and we may refer to these variables as either bits or signals (voltages). To implement multiplication by $[\bar{h}]_i$, the variables b_1 and b_0 are, respectively, the most significant bit and the least significant bit in the binary expression for $[\bar{h}]_i$. Without loss of generality, we may take the filter weights to be integers because $\gamma[\bar{h}]_i$ will always be bounded by one. With this convention,

$$[\bar{h}]_i = 2b_1 + b_0.$$

[0032] Referring now to voltage-to-current converter **502** in Fig. 6, the differential voltage $\{x_+(t), x_-(t)\}$ is applied to input ports **602** and **604**. Current source **602** provides bias current to a differential pair comprising pMOSFETs **608**, **610**, **612**, and **614**. The gates of pMOSFETs **608** and **610** are connected to input port **602**, and the gates of pMOSFETs **612** and **614** are connected to input port **604**. The combination of pMOSFET **608** to pMOSFET **610** looks like a classical cascode pair, save for the fact that the gate of pMOSFET **608** is not at a fixed voltage. A similar statement applies to the combination of pMOSFET **612** and pMOSFET **614**. Accordingly, the voltage-to-current converter of Fig. 6 may be termed an active cascode voltage-to-current converter.

[0033] The betas of pMOSFET **608** and **610** may be chosen such that the active cascode configuration of pMOSFETs **608** and **610** forces pMOSFET **608** to operate in the triode region when pMOSFET **610** is in its active region. A similar statement applies to the combination of pMOSFETs **612** and **614**. This may be observed as follows. Let

V_{S2} denote the source voltage of pMOSFET 610 and V_{S1} denote the source voltage of pMOSFET 608. With pMOSFET 610 in its active region, $V_{S2} > V_g + |V_T|$, where V_g is the gate voltage and V_T is the threshold voltage. (For simplicity, we take the threshold voltage to be the same for pMOSFET 608 and 610.) Simple manipulation of the previous inequality yields $V_{GT} > V_{SD}$, where V_{GT} is defined as $V_{S1} - V_g - |V_T|$ and V_{SD} is the source-drain voltage $V_{S1} - V_{S2}$, which indicates that pMOSFET 608 operates in its triode region.

[0034] With pMOSFETs 608 and 612 operating in the triode region, they act approximately as resistors to degenerate pMOSFETs 610 and 614, respectively.

Degeneration provides a relationship between the voltage difference $x_+(t) - x_-(t)$ and the current difference $I_+ - I_-$ that is linear over a wider range than if degeneration was not present. (I_+ is the drain-source current for pMOSFETs 608 and 610, and I_- is the drain-source current of pMOSFETs 612 and 614, respectively.) This is seen by considering a simple low frequency small-signal T-model for the active cascode voltage-to-current converter of Fig. 6, where it is easily shown that the input-output relationship of the voltage-to-current converter 502 is given by $I_+ - I_- = g_e[x_+(t) - x_-(t)]$, where the effective transconductance g_e is given by $g_e = 1/(r_s + r_L)$, where $r_s = 1/g_m$, the reciprocal of the small-signal transconductance g_m for pMOSFETs 610 and 614, and r_L is the triode region drain-source resistance for pMOSFETs 608 and 612. (For simplicity, we assume that the differential pair is matched.) If pMOSFETs 608 and 612 were not present, then the simple T-model provides $I_+ - I_- = g_m[x_+(t) - x_-(t)]$. Clearly, $g_e < g_m$, so that degeneration lowers the effective transconductance. However, r_s is a function of the input voltage, and it is seen that because $g_e = 1/(r_s + r_L)$, there is less variation in g_e with input voltage than for g_m .

[0035] To simplify the discussion of how multiplication is performed, we may without loss of generality normalize the filter weights to integers. This is so because $\gamma[\bar{h}]_i$ will always be bounded by one. That is, γ absorbs any normalization constant.

With this convention, $[\bar{h}]_i = 2b_1 + b_0$. Referring now to current steering DAC 504 in Fig. 6, varying amounts of current are shunted to ground, depending upon the signals b_1 and

b_0 . In the particular embodiment shown in Fig. 6, pMOSFETs 616 have a channel width-to-length ratio twice that of pMOSFETs 618. When b_1 and b_0 are both a binary 0 (LOW voltage), current from voltage-to-current 502 is shunted to ground by current steering DAC 504 so that the multiplier is zero. When b_0 is 1 and b_1 is 0, currents into nodes 510 and 512 receive 1/3 of the currents from voltage-to-current converter 502 so that the multiplier is 1/3. When b_0 is 0 and b_1 is 1, 2/3 of the currents from voltage-to-current converter 502 is passed through to loads 506 and 508 so that the multiplier is 2/3. When both b_1 and b_0 are 1, all of the currents from voltage-to-current converter 502 are passed through to loads 506 and 508 so that the multiplier is 1. From this discussion considering all the possible values for b_1 and b_0 , it is seen that current steering DAC 504 multiplies the currents from voltage-to-current converter 502 by $\gamma[\bar{h}]_i$.

[0036] The multiplier circuit structure represented by current source 602, voltage-to-current converter 502, and current steering DAC 504 is repeated for each tap weight in FIR 402 of Fig. 4 with the appropriate input voltages. That is, each multiplier has its own current source, voltage-to-current converter, and current steering DAC, where for each $i = 0, 1, \dots, n - 1$, the bit representation for $[\bar{h}]_i$ is provided to the current steering DAC of the i^{th} multiplier circuit and the differential voltage $\{x_+(t-i), x_-(t-i)\}$ is provided to the voltage-to-current converter of the i^{th} multiplier circuit. Extension of the multiplier circuit in Fig. 6 to arbitrary bit representations of the weights should be straightforward. This same circuit multiplier structure may also be used to perform the multiplication indicated by multiplier 408 in Fig. 4, where now the bit representation for the scale factor K is provided to the current steering DAC of multiplier 408 and the differential voltage $\{d_+(t), d_-(t)\}$ is provided to the voltage-to-current converter of multiplier 408.

[0037] The addition operation indicated by summer 410 is implemented by connecting the output ports of each current steering DAC for FIR 402 and multiplier 408 to loads 506 and 508. Which particular loads these output ports are connected to determine the sign of the multiplier. For example, without loss of generality, we may take a positive multiplicative weight to be implemented by connecting the output port in the I_+ current path to load 508 and the output port in the I_- current path to load 506. Then,

to implement multiplication by a negative weight, the output port in the I_+ current path is connected to load 506 and the output port in the I_- current path to load 508.

[0038] The combination of loads 506 and 508 (Figs. 5 and 6) and comparator 412 (Fig. 4) may be realized by the circuit of Fig. 7, which may be referred to as an active cascode differential latch. In Fig. 7, nMOSFETs 714 and 716 play the role of active cascode transistors, although they are not cascode transistors in the classical sense because their gate voltages are not biased to a constant voltage. Input port 702 is connected to node 512 and input port 704 is connected to node 510. A clock signal in Fig. 7 is denoted by φ . When clock signal φ is HIGH, pre-charge transistor 722 provides a low impedance path between the gates of transistors 710 and 712, pre-charging the differential latch so that the output voltages at output ports 706 and 708 are forced to be substantially equal to one another. When clock signal φ is LOW, pre-charge transistor 722 switches OFF to put the differential latch in an evaluation mode, where a differential voltage at output ports 706 and 708 develops in response to the currents (or voltages) at input ports 702 and 704.

[0039] If more current is sourced to input port 702 than is sourced to input port 704, then the output voltage at output port 706 increases, and the output voltage at output port 708 decreases. Cross coupled pMOSFETs 718 and 720 are connected as a latch, so that the differential voltages developed at output ports 706 and 708 are amplified to a complementary logic levels. The resulting complementary voltages may both be used in subsequent digital processing, or only one of the complementary voltages may be used. For example, dual rail logic may be employed in some or all of the subsequent digital processing.

[0040] An argument similar to that which was made with respect to the voltage-to-current converter in Fig. 6 shows that device sizes may be easily chosen such that nMOSFETs 710 and 712 operate in their triode regions and nMOSFETs 714 and 716 operate in their active regions. With nMOSFETs 710 and 712 operating in their triode regions, the differential latch of Fig. 7 has a relatively low input impedance, and furthermore, these nMOSFETs require a relatively small amount of headroom voltage. This may be important as supply voltages are scaled lower and lower, because voltage headroom for analog circuit blocks may be a critical issue. A small headroom voltage

requirement also improves the input common-mode voltage range of the stages previous to the differential latch, i.e., the voltage-to-current converters and current steering DACs.

[0041] As discussed earlier, the scale factor K takes into account various scaling factors due to the communication channel and circuit implementation. The scale factor K affects the available noise margin, so it is important to set its value appropriately. The optimal value for K depends on the communication channel characteristics, which usually are not known a priori. A relatively simple method for calibrating K making use of a received training sequence is shown in the flow diagram of Fig. 8.

[0042] In block 802, K is initialized to zero. In block 804, the filter is updated over a training sequence. During this update, a count is made of the number of overflows in the components of the filter vector. In block 806, the overflow count is compared to a threshold λ . If the overflow count is greater than λ , control is brought to block 808, in which case the current value for K is decremented by Δ and the calibration ends. If the overflow count is less than λ , control is brought to block 810 where K is incremented by Δ , and control is brought to block 804 to begin another update sequence for the filter weights with the new value for K . Block 806 may be modified to shift control to block 808 if the overflow count is greater than or equal to the threshold λ .

[0043] The overflow count threshold λ and scale factor calibration increment Δ are chosen before calibration is performed. These scalars may be determined offline via simulations with expected communication channels. In some embodiments, the overflow count threshold may be set to a value less than 1% of the length of the training sequence. The calibration increment trades off coarseness with the number of times block 804 is performed. In practice, calibration of the scale factor need not be performed very often, and may in some instances be performed only once for a communication channel.

[0044] A hardware implementation for the calibration method of Fig. 8 is shown at a high-level in Fig. 9. Many of the functional units shown in Fig. 9 have equivalents in Fig. 4 and are labeled accordingly with the same numerals. These functional units need not be described again. For simplicity, functional units 404 and 406 in Fig. 4 have been combined into functional unit 902. In addition to the functional units shown in Fig. 4, Fig. 5 also includes overflow counter 904, finite state machine (FSM) 906, multiplexer

908, summer **910**, delay element **912**, and summer **914**. As seen in Fig. 9, calibration of K adds only a modest amount of hardware to implementing the filter update.

[0045] Summer **914** is similar to summer **420** in Fig. 4, but a different numerical label is used to emphasize that summer **914** provides an overflow signal to overflow counter **904** to indicate when a numerical overflow occurs during summation. Overflow counter **904** keeps track of the number of numerical overflows during a training sequence period (block **804**). FSM **906** initializes overflow counter **904** before the beginning of a training sequence, and controls data generator **902** to begin generating a sequence of data symbols when a new iteration on a training sequence begins.

[0046] Functional units **908**, **910**, and **912** perform either block **808** or block **810**. Delay element **912** indicates a delay of L , which denotes the length of a training sequence. If at the end of a training sequence the overflow counter indicates that the overflow count is greater than the threshold λ , FSM **906** sends a control signal to multiplexer **908** so that $-\Delta$ is added to the current value of K for use in the next training sequence. Otherwise, Δ is added to the current value of K . As discussed with respect to block **806**, this procedure may be modified so that $-\Delta$ is added when the overflow count is greater than or equal to the threshold λ .

[0047] The method outlined in Fig. 8 need not be limited to the filter update indicated by equation (1), but may also be used for other filter update algorithms involving a scale factor, such as, for example, the LMS and sign-sign LMS algorithms discussed in the Background section. With this in mind, the hardware diagram of Fig. 9 may be generalized to that of Fig. 10. Referring to Fig. 10, filter increment generator **1002** receives as input the difference $z(t) - Kd(t)$ from summer **410** and the received data vector $\bar{x}(t)$ from FIR **402**, and outputs to summer **914** the increment vector $\delta\bar{h}(t)$ that is to be summed with the current filter vector $\bar{h}(t)$ to arrive at the updated filter vector $\bar{h}(t+1)$. That is, $\bar{h}(t+1) = \bar{h}(t) + \delta\bar{h}(t)$. Data generator **1002** is labeled with a different numeral to indicate that it may or may not provide data information, such as $\text{sgn}\{d(t)\}$, to filter increment generator **1002**.

[0048] Many modifications may be made to the disclosed embodiments without departing from the scope of the invention as claimed below. For example, in the case of

single-ended signaling, one of the input voltages to the various voltage-to-current converters is held constant instead of being part of a differential signal. Furthermore, the circuit structure disclosed here to implement weight multiplication in the analog domain merely serves as an example, and is not meant to limit the scope of the invention in any way. For example, it is to be noted that the channel width-to-length ratios for the pMOSFETs used in the current multiplier need not be powers of two. That is, it is not necessary that multiplication be performed in binary arithmetic. Also, the duals to the disclosed circuits may be implemented, where nMOSFETs replace pMOSFETs, and pMOSFETs replace nMOSFETs. Other circuit structures for performing multiplication and summation in the analog domain may be employed. Indeed, the multiplication and summation may also be performed entirely in the digital domain, although such circuit implementations, which would require analog-to-digital converters, may at this time not be practical for high-speed communication channels.

[0049] The partitioning of a circuit into simpler, functional units is somewhat arbitrary, and the particular functional units disclosed here are not meant to limit the scope of the invention. Various functional units may be combined into more complicated functional units, and functional units may be partitioned into simpler functional units. An example is the combining of functional units 404 and 406 of Fig. 4 into data generator functional unit 902 of Fig. 9. Furthermore, the various summers and multipliers disclosed here may be re-arranged as implied by arithmetic commutative and associative laws. In particular, it is to be understood in the claims below that the term “summer” also may mean a functional unit for performing a differencing operation. Furthermore, “a digital summer” may comprise more than one summer to operate on more than one pair of operands in parallel. Similarly, “a delay element” may comprise more than one delay elements to delay more than one value in parallel. Similar remarks apply to multipliers and latches.

[0050] It is also to be understood in the claims below that the various summers and multipliers, whether realized in the digital domain or analog domain, may perform their indicated operations only approximately. For example, in the analog domain, it is not possible to match transistors exactly, so that a multiplication is not exact. Or, for example, in the digital domain there may be a numerical overflow.

[0051] Regarding calibration of the scale factor K , training sequences for the calibration may be a repetition of a training sequence, or segments of one long training sequence. Furthermore, the increment used in block 808 of Fig. 8 need not be the same increment as that used in block 810, and the increment may be made to vary each time control is brought to block 802.